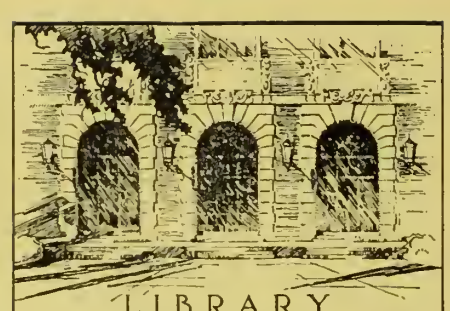


LIBRARY
OF THE
UNIVERSITY
OF ILLINOIS

510.84
I l6r
no. 243-249
cop. 2

The person charging this material is responsible for its return on or before the **Latest Date** stamped below.

Theft, mutilation, and underlining of books are reasons for disciplinary action and may result in dismissal from the University.

UNIVERSITY OF ILLINOIS LIBRARY AT URBANA-CHAMPAIGN

MAR 13 1972

FEB 16 RECD

L161— O-1096

Digitized by the Internet Archive
in 2013

http://archive.org/details/websystem243haze

Report No. 243

COO-1018-1123

no. 243-249

THE WEB SYSTEM
PART I: THE UTILIZATION OF WEB

by

J. H. Hazelhurst
B. H. McCormick
W. D. Bond

THE LIBRARY OF THE
DEC 21 1967
UNIVERSITY OF ILLINOIS

September 11, 1967

DEPARTMENT OF COMPUTER SCIENCE · UNIVERSITY OF ILLINOIS · URBANA, ILLINOIS

Report No. 243

THE WEB SYSTEM
PART I: THE UTILIZATION OF WEB

by

J. H. Hazelhurst
B. H. McCormick
W. D. Bond

September 11, 1967

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

Supported in part by Contract AT(11-1)-1018 with the U.S. Atomic Energy Commission and the Advanced Research Projects Agency.

THE WEB SYSTEM: GENERAL INTRODUCTION

To describe the interconnection and packaging of the logical elements in a contemporary digital processor can be a formidable task. The WEB System is an operating system to be used as a tool by the logic designer in completing his task.

The two main goals of the system are to:

1) relieve the designer from the tedious jobs associated with the design and documentation of a digital processor, i.e., the logical description of the processor, and

2) enable the designer to use previous units of design as well as to easily change the design of a processor still in the design phase.

Furthermore, the ultimate result of the system is the generation of information needed for fabrication of a digital processor. More concisely, the WEB System is an operating system for logical design automation.

This report is divided into four main parts:

Part I: The Utilization of WEB

An example from the design of Illiac II is carried through in detail, showing how to use the WEB language.

Part II: A Formal Description of the WEB Language

This is a formal description of the language used within the system to describe the interconnection and packaging of a digital assembly.

Part III: WEB Change Orders

This is a description of the commands available in the operating system to dynamically change a particular design within the system.

Part IV: Implementation

Information concerning the implementation of the design automation system is given here.

1. INTRODUCTION

This report is an introduction to WEB, a language to describe the interconnection and packaging of the logical elements of a digital assembly.[1] An example from the design of Illiac II is presented in an elementary manner to show how the various constructs of the language can be used. Printed circuit fabrication is discussed for illustrative purposes. However, the language is useful for any type of pluggable unit, for example, in integrated circuit technology.

It is to be noted that only examples of usage are given in this report. A precise definition of the language may be found in Part II of the WEB System: A Formal Description of the WEB Input Language, Department of Computer Science Report No. 232, June 19, 1967. It is recommended that the user familiarize himself with the concepts of the language found herein before burdening himself with the exact formats which the input must take as given in the formal description.

The input data necessary to create a WEB file is divided into four parts:

1) The Logic Description specifies the interconnection of the logical elements of the system.

2) The Housing Description specifies the physical characteristics of the circuit modules, printed circuit card connectors, card drawers, etc.

3) The Module Description specifies the logical makeup of a pluggable unit.

4) The Assignment Description specifies the assignments of pluggable units to locations in the housing, and logic units to locations on the pluggable unit.

Drawing D-3153 for Illiac II is used as an example to show how these descriptions are to be given to the WEB processor. (See Figure 1)

---

[1]The description of WEB as given here is a modification and generalization of concepts found in: J. H. Hazlehurst and B. H. McCormick, WEB: A Language to Describe the Interconnection and Packaging of the Logical Units in a Digital Processor, Part I: Preparation of Input Data, DCS File No. 625, Revised February 10, 1965.

2. INPUT FORMATS

The WEB language uses what is known as a "free field" format. It is not necessary to begin a statement in any particular column of a card, nor is it necessary for a statement to be entirely on one card. Rather, the input is considered as simply a stream of characters. Blanks may be inserted anywhere within this stream except in reserved words, such as JØB, TITLE, etc. Comments may also be inserted anywhere in the character stream. A comment starts with the characters /* and ends with the characters */. Anything between the starting and ending symbols is treated as comment and is printed but otherwise ignored.

Although a free format is used, the input is listed card by card. The user may decide to start in a particular column to improve legibility. It is highly desirable to use indentation to clarify the structure of nested FOR loops.

3. SPECIAL OPERATIONS

The operations listed in this section are used to document the listing as well as output, and to improve legibility. None of these operations need be present in a particular WEB run, and their use is left entirely to the discretion of the user.

3.1 TITLE Statement

This operation is used to cause a title line to be printed at the top of every page of the listing of the input. When the operation is encountered, a new page is started and the title statement is the first card listed on the new page. The actual title itself must be less than 60 characters in length.

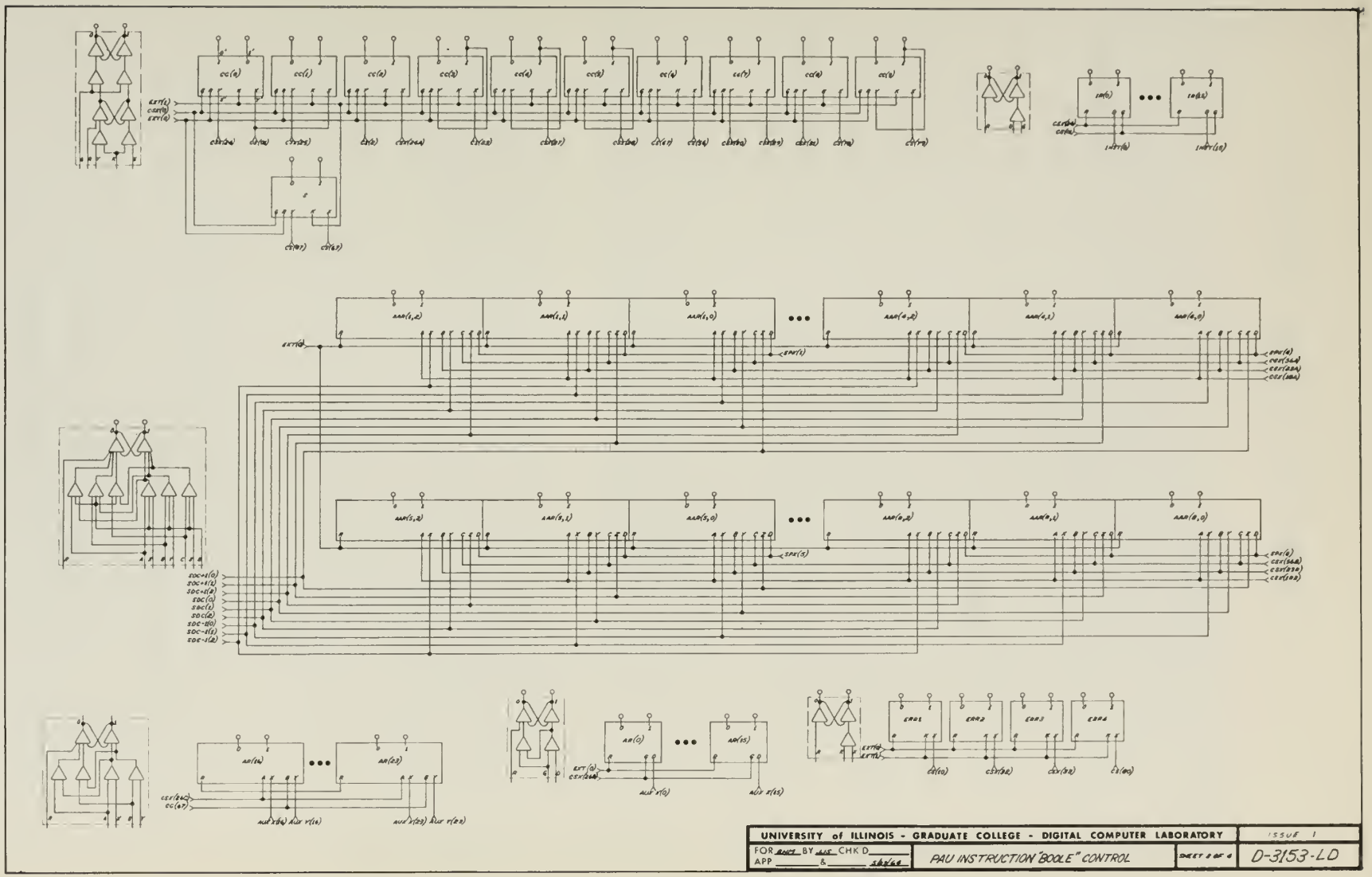
3.2 DRAWING Statement

This operation is used to associate the input data with a particular drawing. The information provided will be used to compile a title-drawing dictionary, and to reference logical elements to the drawings in which they appear. All input cards following a drawing statement apply to the most recent drawing statement read.

3.3 JOB Statement

This operation is similar to the TITLE statement except that the title information is printed at the top of every page of output generated by the WEB processor. If a JOB card is present, it must be the first card in the input deck.

FIGURE 1



UNIVERSITY of ILLINOIS - GRADUATE COLLEGE - DIGITAL COMPUTER LABORATORY		ISSUE 1
FOR ____ BY ____ CHK'D ____ APP ____ & ____	PAU INSTRUCTION "BOOLE" CONTROL	SHEET # OF # / D-3153-LD

4. PREPARATION OF THE LOGIC DESCRIPTION

4.1 Preliminary Information

Perhaps the best explanation of the preparation of input data is by way of example. To this end, we shall develop the logic description of Illiac II Drawing D-3153 as given in Figure 1.

The first piece of input is the JOB statement (See Figure 2). The title "OUTPUT FRØM SAMPLE INPUT TO THE WEB PROCESSOR" will appear at the top of every page of output produced by the WEB processor. Since blanks in no way affect the input, we leave a couple of blank lines to increase readability, and then insert a comment which indicates the purpose of this input deck.

The first statement of any real significance is the LOGIC statement which indicates to the WEB processor that a logic description is next in the input deck.

This statement must precede the rest of the data in the LOGIC section of the input. In any one job the logic description must be given in one consecutive deck. It may not, for example, be split with a piece of the HOUSING description in between.

Following this is a TITLE statement. This provides additional titling for the listing of the input data itself. A new page will be started and the title "PAU "BOOLE" CONTROL LOGIC" will be printed. The listing then continues with the image of the TITLE card.

Next is a DRAWING statement indicating with which drawing the input is to be associated.

```
JOB,   /* OUTPUT FROM SAMPLE INPUT TO THE WEB PROCESSOR */

/*  THIS IS A SAMPLE INPUT TO WEB */
                                    ;
    LOGIC ;
    TITLE, /* PAU "BOOLE" CONTROL LOGIC */  ;
    DRAWING, D-3153     /* "BOOLE" CONTROL*/   ;
```

FIGURE 2

4.2 Basic Units

We now begin describing the actual logic as indicated on the drawing (See Figure 3). We must start with the basic logical units, which in this case are NAND elements. Once these basic units are described, they may be assembled into blocks (such as those labeled "CC" and "AAR") which in turn may be assembled into yet larger blocks. These blocks must be declared to the WEB processor in order of increasing complexity. For this reason we must start with the basic logical units.

The first part of the description of a logical unit is the name by which the user intends to refer to the particular unit, followed by a colon. Then the word UNIT, followed by a comma, indicates to the processor that we are defining a basic logical unit. Next, we give the names of the terminals of the unit being specified. These names may be completely arbitrary--they serve only to reserve a "space" where one end of a wire may later be attached. The terminal names have no significance whatsoever external to the unit description. Thus, the same names may be used to name terminals of distinct units, as in Figure 3.

Although the actual names of the terminals are immaterial, the position in the terminal list is very important. For example, we shall use the first two positions in the terminal list of NAND1 as inputs to the element, and the last position as the output. It is essential that every time we use NAND1, we remember that the last position corresponds to the output of the element. This is just a convention, as we could have just as easily used the first position as the output--the only requirement is consistency in usage.

```
/*  THE FOLLOWING ARE THE BASIC UNITS  */
    NAND1 :  UNIT,  A,B,C          /* 2 INPUT NAND */  ;
    NAND2 :  UNIT,  A,B,C,D        /* 3 INPUT NAND */  ;
    NAND3 :  UNIT,  A,B,C,D,E      /* 4 INPUT NAND */  ;
    NAND4 :  UNIT,  A,B,C,D,E,F    /* 5 INPUT NAND */  ;
```

FIGURE 3

4.3 Blocks

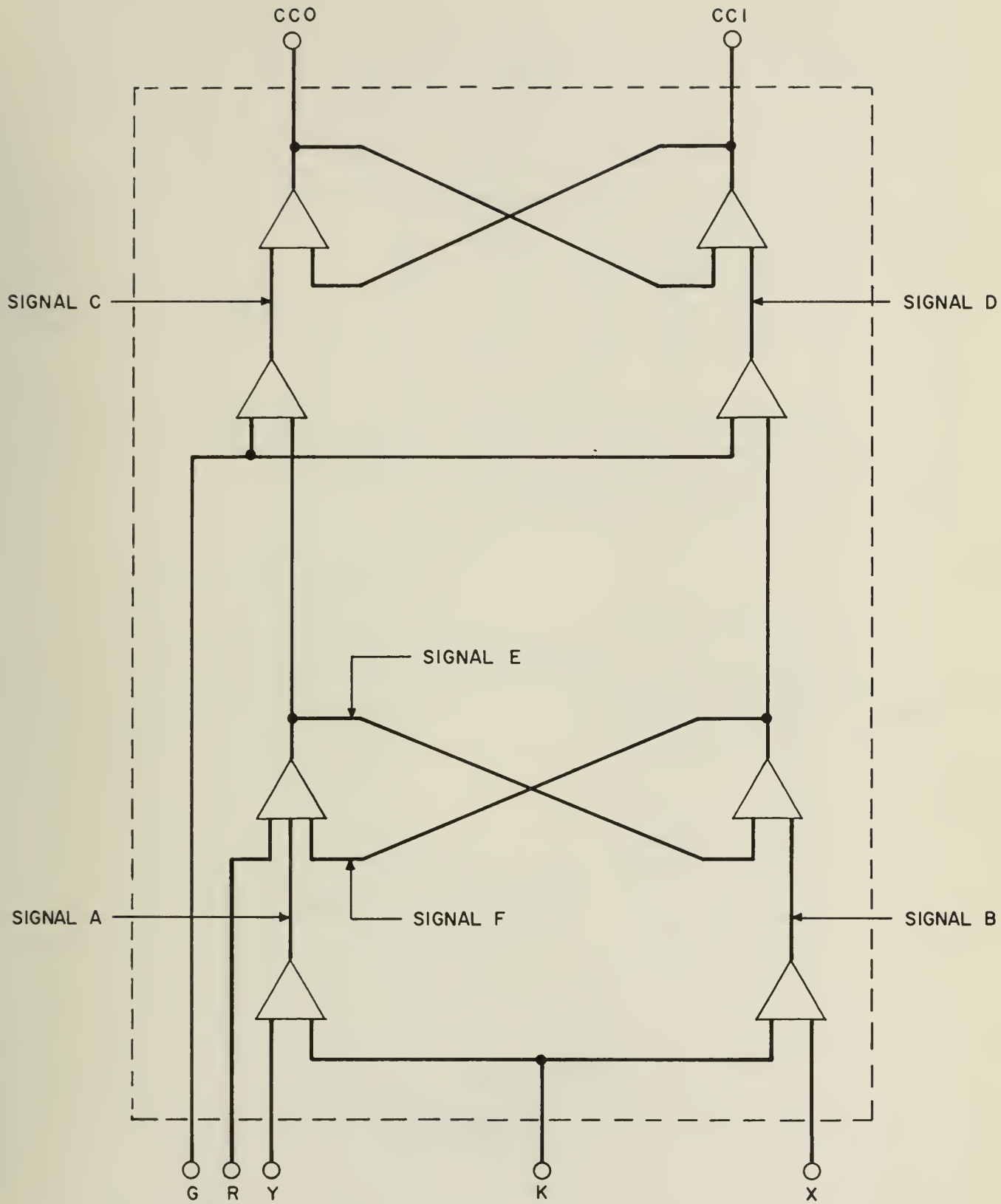
Once the basic units have been described, we may use them to build more complex logical entities which we shall call blocks. For example, in Figure 1 we observe pieces of logic outlined by broken lines. These blocks are not explicitly named on the drawing, but their proximity to assemblies of blocks with names indicates that the names used there extend to these broken line blocks.

As an example to show how a block is described in WEB, we shall use the block in the upper left corner of Figure 1, which we shall name CC.

Our first task is to declare to the processor that we are describing a block name CC, and to indicate the names of the terminals of the block. This is done in the same way as a unit is described except that the word BLØCK is used rather than UNIT (See Figure 4). In this case, the input terminals are already named G, R, Y, K, and X, while the output terminals are just labeled 0 and 1 on the drawing. Since every terminal name must begin with a letter, we shall call the output terminals CC0 and CC1.

Next we must indicate to the processor the logical structure of the block, by describing the interconnection of the basic units (which we have already described to the processor) within the block. This block is composed of seven two-input (NAND1) nands and one three-input (NAND2) nands. To be able to describe the interconnection of these elements, we must give a name to each signal of the block. Since the terminal signals have already been given names, we only need to name those signals within the block which are not terminal signals. It is essential that the names we use have not already been given to terminal signals, but otherwise the choice of names is immaterial.

The following sketch shows block CC with all signals named.


CCO
CCI
SIGNAL C
SIGNAL D
SIGNAL E
SIGNAL A
SIGNAL F
SIGNAL B
G
R
Y
K
X

After naming each signal, we indicate to the processor how the block is to be constructed. This is done by giving the name of each (previously defined) basic unit along with the names of signals being input to and output from the unit. One does this for each unit in the block, and thus specifies the interconnection of the units comprising the block. For example (See Figure 4), the entry for the nand in the upper left corner of the above sketch is:

NAND1, C, CC1, CCO

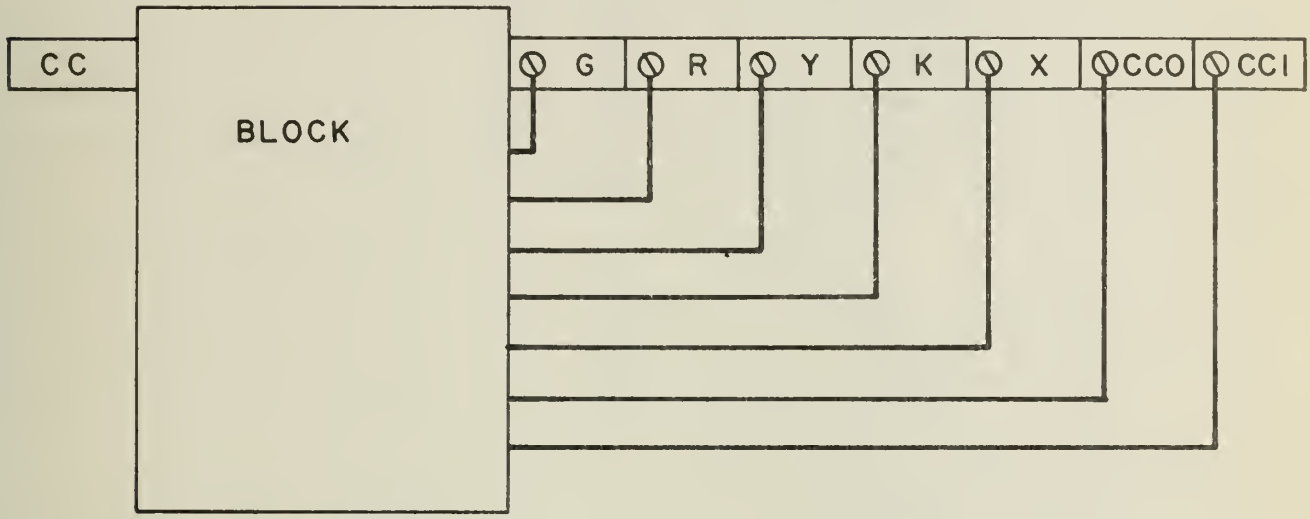
indicating that signals C and CC1 are inputs to the nand, and CCO is the output signal.

After an entry has been made for each unit of the block, we indicate the end of the block description by using the command END which may be followed by the name of the block if we so desire.

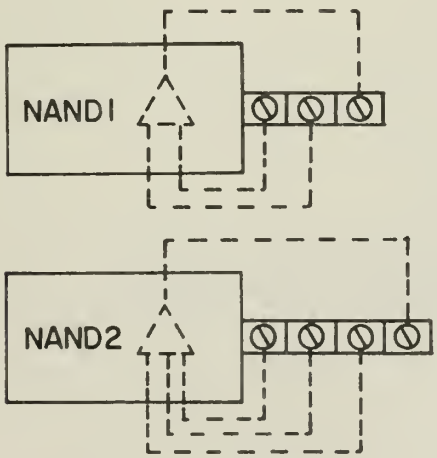
```
/*  FIRST LEVEL BLOCKS */
CC : BLOCK,   G,R,Y,K,X,CCO,CC1 ;
     NAND1,   C,CC1,CCO ;
     NAND1,   CCO,D,CC1 ;
     NAND1,   G,E,C ;
     NAND1,   G,F,D ;
     NAND2,   R,A,F,E ;
     NAND1,   E,B,F ;
     NAND1,   Y,K,A ;
     NAND1,   K,X,B ;
     END,  CC ;

S :  EQU,  CC ;
```

FIGURE 4

The following visualization may help. Imagine the block as follows: from the left side sticks a sign with the name of the block on it; from the right side sticks a "terminal strip." A terminal is provided for each named signal (wire)entering or leaving the block:



We now indicate to the processor what is inside the block and how these parts are hooked together. We have previously defined the <u>units</u> involved:



We now arrange these units inside the block and the connecting wires.

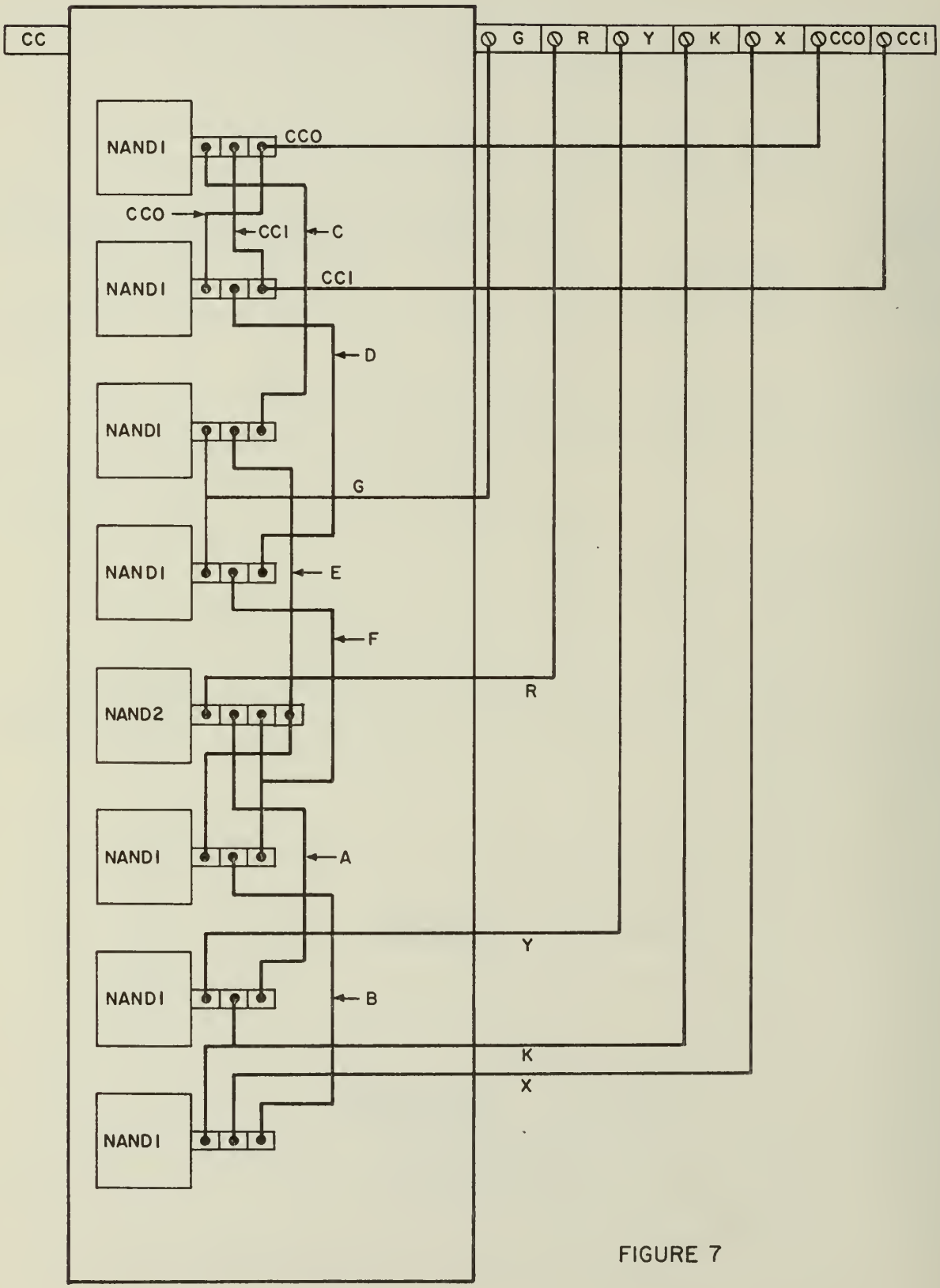


FIGURE 7

If two blocks are identical in structure, yet it is desired to give the blocks different names, it is only necessary to give the complete description of one of the blocks. The other may be described by the use of an EQU instruction. For example, in Figure 4 the block named S is described as having the same structure as block CC. The drawing indicates that this is exactly the structure we wish block S to have.

Looking once again at the drawing, we see two different blocks both of which are named "AR". This is not allowed by the WEB processor, since names are used to identify specific blocks. If a new block is given a name which has already been used, a message to this effect will be printed, and the new description will replace the old one. From this point on, all references to the redefined name will cause insertion of the new form of the block. In this particular instance, we may resolve the ambiguity by naming the blocks "AR1" and AR2".

Rather than continue to define all first level blocks, we turn now to using the first level blocks in the description of second level blocks--the most complex logical networks on the drawing. To insure that the reader understands the procedure of defining blocks, he may describe the blocks labeled IR, AR1, AR2, or AAR, the descriptions of which are correctly given in Figure 8(of course, there may be many correct descriptions of a given block, those in Figure 8 are just particular correct descriptions).

To describe higher level blocks, we just use previously defined blocks as pseudo "units", in much the same manner as we described first level blocks using units. There is a slight difference however. The second level blocks of this drawing have not been assigned names, and they do not appear to have well defined "terminal strips".

The problem concerning names is easily set aside by the following rule: If a block is to be used in a higher level block description, the block must have a name by which it can be called. If a block is not named, it cannot be used to build more complex blocks.

```
IR :  BLOCK,   R,D,G,IRO, IR1 ;
      NAND1,   R,IR1, IRO  ;
      NAND1,   IRO,A,IR1 ;
      NAND1,   D,G,A ;
      END;

AR1:  BLOCK,   R,G,D,AR10,AR11 ;
      NAND2,   R,A,AR11,AR10 ;
      NAND1,   AR10,B,AR11 ;
      NAND1,   G,B,A ;
      NAND1,   G,D,B ;
      END;

AR2:  BLOCK,   R,A,X,B,Y,AR20,AR21 ;
      NAND3,   R,L,M,AR21,AR20 ;
      NAND2,   AR20,N,O,AR21 ;
      NAND1,   A,N,L ;
      NAND1,   B,O,M ;
      NAND1,   A,X,N ;
      NAND1,   B,Y,O ;
      END,  AR2 ;
AAR:  BLOCK,   R,A,X,B,Y,C,Z,D,AAR0,AAR1 ;
      NAND4,   R,E,F,G,AAR1,AAR0 ;
      NAND3,   AAR0,H,I,J,AAR1 ;
      NAND2,   A,D,H,E ;
      NAND2,   B,D,I,F ;
      NAND2,   C,D,J,G ;
      NAND2,   A,D,X,H ;
      NAND2,   B,D,Y,I ;
      NAND2,   C,D,Z,J ;
      END,  AAR ;
```

FIGURE 8

An unnamed block cannot have a terminal list. In effect, all signal names given in the description of an unnamed block are made into a terminal list, for the processor makes all signal names inside an unnamed block available to the outside. Then if another unnamed block appears anywhere in the logical description, signals in each unnamed block with the same name are assumed to be the same signal. This convention makes all unnamed blocks into one large all encompassing block, pieces of which are defined at various places in the input.

Figure 9 shows definition of an unnamed block using previously defined blocks CC and S in its body of definition.

```
/* SECOND LEVEL BLOCKS */

BLOCK;

CC,   CSX(0), EXT(0), CSX(24), EXT(1), CS(1) , CCO(0), CC1(0) ;
CC,   CSX(0), EXT(0), CSX(25), EXT(1), CS(1) , CCO(1), CC1(1) ;
CC,   CSX(0), EXT(0), CS(2) ,  EXT(1), CSX(26,1),CCO(2),CC1(2) ;
CC,   CSX(0), EXT(0), CC1(3),  EXT(1), CS(23), CCO(3), CC1(3) ;
CC,   CSX(0), EXT(0), CC1(4),  EXT(1), CSX(27),CCO(4), CC1(4) ;
CC,   CSX(0), EXT(0), CC1(5),  EXT(1), CSX(28),CCO(5), CC1(5) ;
CC,   CSX(0), EXT(0), CS(67),  EXT(1), CS(56), CCO(6), CC1(6) ;
CC,   CSX(0), EXT(0), CSX(30), EXT(1), CSX(29),CCO(7), CC1(7) ;
CC,   CSX(0), EXT(0), CSX(31), EXT(1), CS(78), CCO(8), CC1(8) ;
CC,   CSX(0), EXT(0), CC1(9),  EXT(1), CS(79), CCO(9), CC1(9) ;
S ,   CSX(0), EXT(0), CS(87),  EXT(1), CS(67), SO,     S1 ;
END;
```

FIGURE 9

Each logic description must have at least one unnamed block occurring in it. When we defined named blocks, the signal names in the terminal lists were just dummy names, only serving to reserve a position for connections. However, in an unnamed block, the signals are given names which are not dummy names, which we call actual names. The WEB processor, after all the logic description has been processed, starts with the first unnamed block and replaces each call of a named block in the description by the definition of that block with actual names substituted in place of dummy names. This process continues until all calls of named blocks have been replaced by their definition with actual names substituted for dummy names. The expanded logic description is then printed out in what is called the packaging Skeleton.

4.4 The FOR Clause

A time saving device is the FOR clause, to be used when describing a logical assembly which has an iterative structure. A glance at the IR blocks in the logical drawing will clarify the concept. Instead of explicitly showing every IR block in this simple repetitive arrangement, a ". . ." is used to indicate what happens. The WEB processor will provide for this simple iteration by using a FOR clause as indicated in Figure 10. Each FOR clause must have a matching END.

In the description of the unnamed block composed of AAR's, a more complex usage of FOR clauses is found--that of nested FOR loops. In the original drawing the subscripts "23A" and "23B" occur. This is not allowed by the WEB processor, since only integers may be used as subscripts. Therefore, in the WEB description we must replace the letters with a numbering scheme--in Figure 10 we use multiple subscripting to get around this restriction. Changes of this sort should be noted on the drawing, and descriptive comments should be inserted in the WEB description.

The size of this latter description compared with the amount of information found in the drawing shows clearly the utility of the FOR clause whenever any iterative property is present in the logic design.

The K subscript here corresponds to the two major cycles indicated by signals named CSX(56A), CSX(23A), etc. and the "B" versions of these. The J and I subscripts correspond to the subscripts associated with each AAR occurrence on the drawing. The reader should satisfy himself that this is a correct description of the logic given in the drawing.

```
BLOCK;
  FOR I := 0 UNTIL 15 DO
    IR,   CSX(24), INST(I), CS(1), IRO(I), IR1(I) ;
  END;
END;

BLOCK;
  FOR K := 1 UNTIL 2 DO
    FOR J := 1 UNTIL 4 DO
      FOR I := 2 STEP -1 UNTIL 0 DO
        AAR,   EXT(0), CSX(18,K), SDCM1(I), CSX(23,K),
               SDC(I), CSX(56,K), SDCP1(I), SPX(J+4*(K-1)),
               AARO(J+4*(K-1),I), AAR1(J+4*(K-1),I) ;
      END;
    END;
  END;
END;
```

FIGURE 10

5. PREPARATION OF THE HOUSING DESCRIPTION

In the Logic Description we specified how the basic units of logic were to be interconnected. In the Housing Description we must specify where the terminals of these units are to be located in the actual assembly. Together these two sets of information provide the WEB processor with sufficient information to permit production of wiring tables, logic tracing tables, and other bookkeeping information.

5.1 Connector Descriptions

As in the case of the Logic Description, where we defined blocks of logic by means of previously defined blocks or units, in the Housing Description the same procedure can be used to define connectors in terms of previously defined connectors. As in the case of logic blocks, a connector must be named if it is to be used in the definition of more complex connectors.

In the simplest case, a connector description may be used to define a two-dimensional array of pins. As an example of such a connector description, in Figure 12 the description is given of the following connector, as used in the fabrication of Illiac III.

We name this connector "13 CONN" and the word "PINS" indicates that we are describing a connector. The number between two pin names is the separation factor between these pins. Once such a separation factor has been given, it is in effect as the (horizontal) distance between adjacent pins until a new separation factor is given, and thus it is not necessary to give the separation factor between every two pins. The number between semicolons is the separation factor between rows of the connector. It too remains in effect until a new row separation factor is encountered.

Note that we have given different pins the same name. There are two holes for each pin in the Illiac III connector, and the appropriate distances must be given to indicate this. However, in referring to these

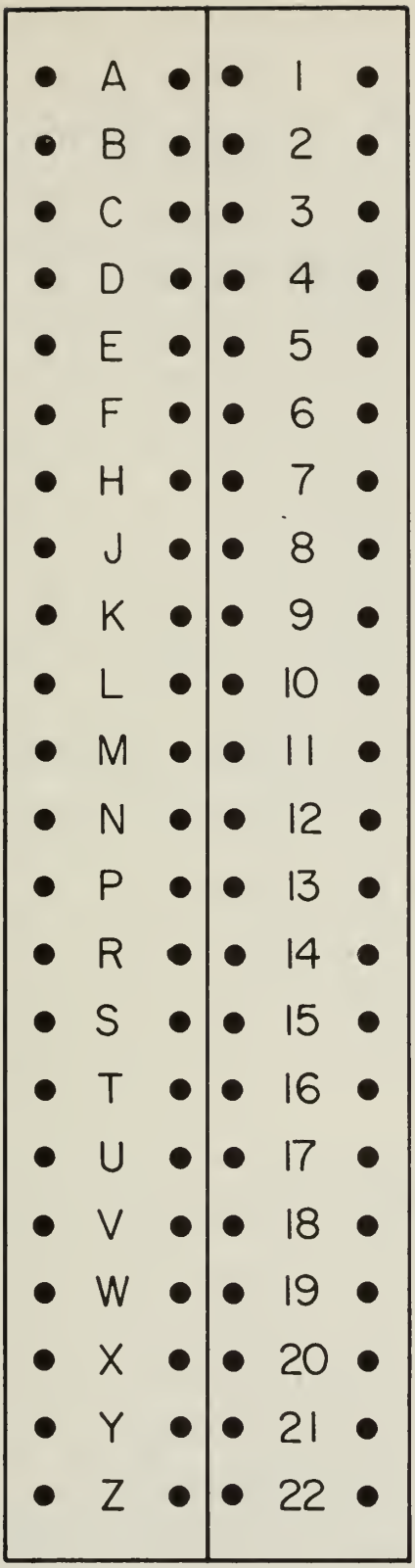

A
B
C
D
E
F
H
J
K
L
M
N
P
R
S
T
U
V
W
X
Y
Z
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22

FIGURE 11

```
HOUSING;
/* THE BASIC CONNECTOR */
I3CONN  :   PINS;                                   ;
            A, /0.094/ , A, 1, 1;
            /0.155/ ;
            B,B,2,2;
            C,C,3,3;
            D,D,4,4;
            E,E,5,5;
            F,F,6,6;
            H,H,7,7;
            J,J,8,8;
            K,K,9,9;
            L,L,10,10;
            M,M,11,11;
            N,N,12,12;
            P,P,13,13;
            R,R,14,14;
            S,S,15,15;
            T,T,16,16;
            U,U,17,17;
            V,V,18,18;
            W,W,19,19;
            X,X,20,20;
            Y,Y,21,21;
            Z,Z,22,22;
        END, I3CONN;
```

FIGURE 12

at a later time we do not want to distinguish between them as they are to be electrically common. Thus, physically we must indicate that there are in fact two holes, but logically we want to treat them identically, so we give them the same name.

At the next higher level of description is the rack, composed of 26 connectors placed in a horizontal row. The connectors in a rack are sequentially numbered, as follows:

1 2 3 4 . . .

A description of this physical arrangement is given in Figure 13. (See below)

A number of racks may be placed one a top another to form a single assembly called a bay. Each individual rack in a bay is identified by a letter:

A
B
C
D

A description of a bay is given in Figure 14. (See following page.)

```
/* DEFINITION OF A RACK */

I3RACK : PINS;
         I3CONN: 1 ; /0.563/ ;
         FOR I := 2 UNTIL 26 DO
             I3CONN : O+I;
         END;
END, I3RACK ;
```

FIGURE 13

```
/*  DEFINITION OF A BAY  */
I3BAY :  PINS;
        I3RACK : A;  /4.438/;
        I3RACK :B ;
        I3RACK : C ;
        I3RACK : D ;
        I3RACK : E;
END,  I3BAY ;
```

FIGURE 14

The components of Figures 13 and 14 essentially consist of two parts. The first of these is the name of the physical connector to be used which has previously been defined. This is followed by a colon and then by what is called an array nomenclature expression. The array nomenclature expression serves to give a logical name to a physical connector, and in turn allows one to give a unique name to each pin of the assembly. This is done by concatenating the nomenclatures of the connectors, of progressively lower orders, until the pin nomenclature has been uniquely identified. For example, pin name A-1-22 would refer to rack A, connector 1, pin 22 of a bay of the Illiac III backpanel.

As in the Logic Description, FOR clauses may be used whenever iterative descriptions are present.

5.2 Cell Descriptions

In the Logic Description, a block could have both external signal names, as given in the terminal list, and internal signal names. When the block was used in a higher level description, the external signal names were changed but the internal signal names were not affected. To allow the same sort of thing in the Housing Description, one must use the CELL.

A CELL is composed of two connectors: an external connector and an internal connector. The connector descriptions of the last section were used in the role of an external connector. It is this connector which is used by higher order structures using the CELL in their construction. In addition to the external connector, the internal connector must be described, and a part of this internal connector must be identified with the external connector by having the same nomenclature. This is similar to the usage of a signal name in the terminal list within the definition of a block.

An example of a cell description is given in Figure 15 where the cabling between two bays is described by making the bays an internal connector and the cabling an external connector. The name following CELL is the name of the external connector of the cell, which as defined in Figure 15 is the first connector of rack A of a bay.

```
WINGCABLE : PINS ;
            I3CONN : A1 ;
END, WINGCABLE ;


I3WING : CELL, WINGCABLE ;
         I3BAY :BAY0 ;
         I3BAY : BAY1 ;
END, I3WING ;
```

FIGURE 15

It is possible that a CELL not have a specified external connector. In this case, the CELL is treated in much the same manner as a block without a terminal list.

6. PREPARATION OF THE MODULE DESCRIPTION

Next we must specify the available modules, or pluggable units. In this example, circuit boards are used, in particular Illiac III board 1018-180. Logically, this board appears as follows:

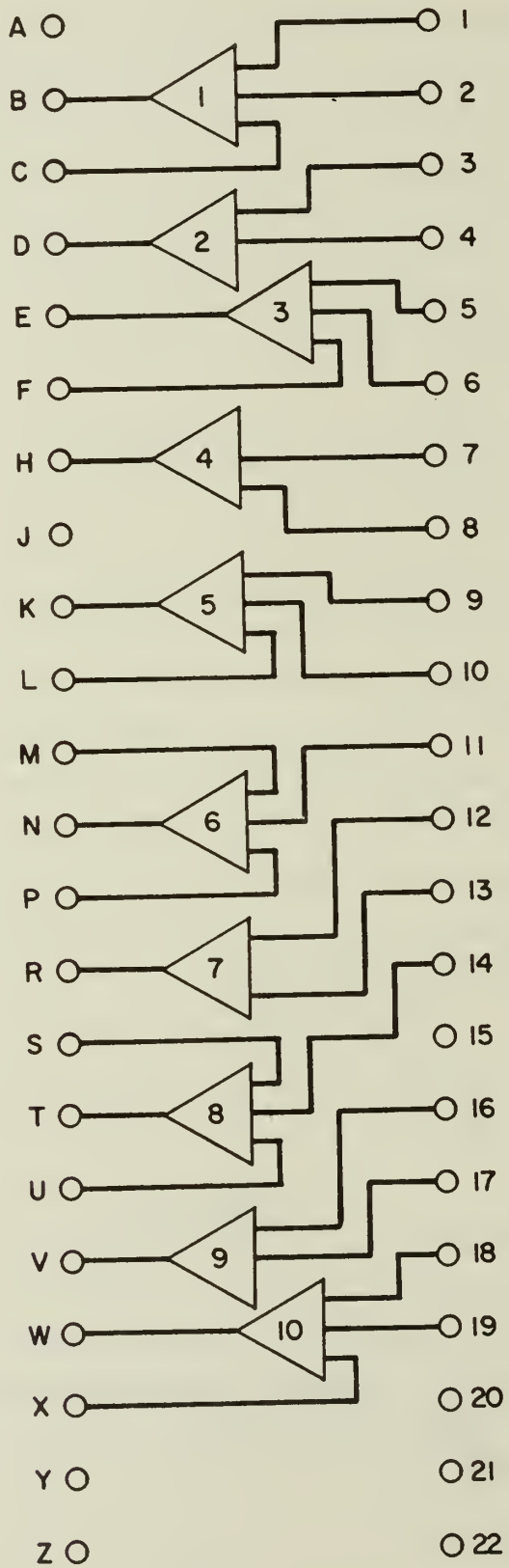


FIGURE 16

We have previously defined (in the Logic Description) the units, NAND2 and NAND1, which appear on this board as must be done. The description of the board is given in Figure 17, where we have named the board A180. For each logical unit on the board, an entry must be made which matches the corresponding definition of the basic unit in the Logic Description, using the correct pin names in the terminal list. The order of the pin names in the terminal list is very important for if the last name in the terminal list has consistently been used as the output pin in the Logic Description then the corresponding pin name must be given last in the entry in the Module Description.

One further thing is done by the WEB processor concerning a module description. Each entry of a module is assigned a sequential number, called the unit's module position, as indicated by the unit symbols in Figure 16.

```
PLUGGABLE UNITS ;
 A180 : MODULE ;
        NAND2, 1,2,C,B;
        NAND1, 3,4,D;
        NAND2, 5,6,F,E;
        NAND1, 7,8,H;
        NAND2, 9,10,L,K;
        NAND2, M,11,P,N;
        NAND1, 12,13,R;
        NAND2, S,14,U,T;
        NAND1, 16,17,V;
        NAND2, 18,19,X,W;
 END,  A180 ;
```

FIGURE 17

7. PREPARATION OF THE ASSIGNMENT DESCRIPTION

So far we have defined the logic of an assembly, the physical characteristics of the housing, and the modules available for fabrication. It is now our task to specify the relationships between these definitions by means of the Assignment Description, composed of a Module Assignment and a Unit Assignment. The Assignment Description thus serves to bind together the logic and packaging of a digital assembly.

7.1 Module Assignment

We must first specify which hardware modules are to be assigned to ("plugged into") the first order connectors defined in the Housing Description. In the example of Figure 18 we have only the module A180 available, and it is assigned to connectors 1 through 10 of racks D and E. All first order connectors must have modules assigned in this manner.

```
        /* SAMPLE BOARD ASSIGNMENT  */

ASSIGN;
A180,   D1,D2,D3,D4,D5,D6,D7,D8,D9,D10 ;
A180,   E1,E2,E3,E4,E5,E6,E7,E8,E9,E10 ;
END;
```

FIGURE 18

7.2 Unit Assignment

Next comes the rather tedious task of assigning existing module units to each occurrence of their corresponding basic logical units of the Logic Description. The Unit Assignment is always entered into the WEB processor in a separate run from the other descriptions. It is anticipated that this section will later be automated, but at present this assignment must be explicitly presented.

Figure 19 shows a section of the full expansion of the Logic Description in which every logical unit in the description is explicitly printed. The numbers down the left hand side of the page, called index numbers, serve to uniquely identify each block or unit occurring in the Logic Description. For example, the third NAND1 of the first block listed here has index number $0103_{(8)}$. Each index number not ending in 00 is the number of a basic unit which must be assigned to a module unit.

This assignment is made as in Figure 20, where the index number is associated with the description of a first order connector and unit's module position. The printed Packaging Skeleton (Figure 19) can be used as a coding sheet for this assignment, entering the physical location in the column headed "LOCATION".

WEB - PASS 1 OUTPUT FROM SAMPLE INPUT TO THE WEB PROCESSOR PAGE 2
PACKAGING SKELETON DRAWING D-3153

INDEX	LOCATION	NAME	SIGNAL 1	SIGNAL 2	SIGNAL 3	SIGNAL 4	SIGNAL 5
0100		CC	CSX(0) CC0(0)	EXT(0) CC1(0)	CSX(24)	EXT(1)	CS(1)
01		NAN01	C	CC1(0)	CC0(0)		
02		NAN01	CC0(0)	0	CC1(0)		
03		NAN01	CSX(0)	E	C		
04		NAN01	CSX(0)	F	0		
05		NAN02	EXT(0)	A	F	E	
06		NAN01	E	B	F		
07		NAN01	CSX(24)	EXT(1)	A		
10		NAN01	EXT(1)	CS(1)	B		
****							
0200		CC	CSX(0) CC0(1)	EXT(0) CC1(1)	CSX(25)	EXT(1)	CS(1)
01		NAN01	C	CC1(1)	CC0(1)		
02		NAN01	CC0(1)	0	CC1(1)		
03		NAN01	CSX(0)	E	C		
04		NAN01	CSX(0)	F	0		
05		NAN02	EXT(0)	A	F	E	
06		NAN01	E	B	F		
07		NAN01	CSX(25)	EXT(1)	A		
10		NAN01	EXT(1)	CS(1)	B		
****							

FIGURE 19

```
/* SAMPLE  FILL*/

FILL;

0101,  D1,7;
0102,  D2,7;
0103,  D1,9;
0104,  D2,9;
0105,  E2,3;
0106,  E1,2;
0107,  E2,2;
0110,  E2,4;

0201,  D3,7;
0202,  D4,7;
0203,  D3,9;
0204,  D4,9;
0205,  E4,3;
0206,  E3,2;
0207,  E4,2;
0210,  E4,4;

END;
```

FIGURE 20

JAN 8 1968

UNIVERSITY OF ILLINOIS-URBANA
3 0112 004249840